Exact synchronization in partial deterministic automata

IOP Conf. Series: Journal of Physics: Conf. Series **1352** (2019) 012047 doi:10.1088/1742-6596/1352/1/012047

Exact synchronization in partial deterministic automata

H Shabana

Institute of Natural Sciences and Mathematics, Ural Federal University, Ekaterinburg, Russia
Faculty of Electronic Engineering, Menoufia University, Menouf, Egypt

E-mail: hananshabana22@gmail.com

Abstract. An automaton is a synchronizing if it has an input word that transfers it from any state to a particular state. There are two versions of synchronization in partial deterministic automata: careful synchronization and exact synchronization. In this paper we focus on the exact version; we survey the complexity of testing exact synchronization and describe a SAT solver based algorithm for calculating the length of the shortest exact synchronizing word.

1. Introduction

A *partial finite automaton* (PFA) $\mathcal{A}$ is a triple $\langle Q, \Sigma, \delta \rangle$, where Q and Σ are finite sets called the *state set* and the *input alphabet*, respectively, and $\delta: Q \times \Sigma \to Q$ is a partial function. The elements of Q and Σ are called *states* and *letters*, respectively, and δ is referred to as the *transition function*.

Let Σ^* stand for the free monoid over Σ, where the empty word ε is the identity element and let $\mathcal{P}(Q)$ be the power set of Q. The transition function δ extends to a function $\mathcal{P}(Q) \times \Sigma^* \to \mathcal{P}(Q)$, still denoted δ, in the following inductive way: for every subset $S \subseteq Q$ and every word $w \in \Sigma^*$, we set

$$\delta(S, w) := \begin{cases} S & \text{if } w = \varepsilon, \\ \{\delta(q, a) \mid q \in \delta(S, v)\} & \text{if } w = va \text{ with } v \in \Sigma^* \text{ and } a \in \Sigma. \end{cases}$$

(The set $\delta(S, v)$ in the right-hand side is defined by the inductive assumption since the word v is shorter than w.) Observe that the set $\delta(S, w)$ may happen to be empty, in which case we say that w is *undefined* at S. In particular, if $\delta(S, w)$ is empty and S consists of a single state q, we say that w is *undefined* at q; otherwise w is said to be *defined* at q.

When we deal with a fixed PFA $\mathcal{A}$, we write $q.w$ for $\delta(q, w)$ and $S.w$ for $\delta(S, w)$.Accordingly, we may simplify the notation for $\mathcal{A}$, writing $\mathcal{A} = \langle Q, \Sigma \rangle$ rather than $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$. An automaton $\langle Q, \Sigma \rangle$ is said to be *complete* if $|q.a| = 1$ for all $(q, a) \in Q \times \Sigma$. We use the acronym CFA for the expression "complete finite automaton".

A CFA $\mathcal{A} = \langle Q, \Sigma \rangle$ is called *synchronizing* if it possesses a word $w \in \Sigma^*$ whose action leaves the automaton in one particular state no matter at which state in Q it is applied: $q.w = q'.w$ for all $q, q' \in Q$. Any w with this property is said to be a *synchronizing* word for the automaton. We refer the reader to the

survey [1] and the chapter [2] of the forthcoming "Handbook of Automata Theory" for a discussion of the rich theory of complete synchronizing automata as well as their diverse connections and applications.

In the literature, there are two basic extensions of the concept of synchronization to PFAs: *careful synchronization* and *exact synchronization*. Careful synchronization has been studied in depth by Martyugin [3–7]. A PFA $\mathcal{A} = \langle Q, \Sigma \rangle$ is said to be *carefully synchronizing* if there is a word $w = a_1 \dots a_\ell$, with $a_1, \dots, a_\ell \in \Sigma$, that satisfies the following conditions $(C1) - (C3)$:

$(C1)$: the letter a_1 is defined at every state in Q,
$(C2)$: the letter a_t with $1 < t \leq \ell$ is defined at every state in $Q.a_1 \dots a_{t-1}$,
$(C3)$: $|Q.w| = 1$.

Any word w satisfying $(C1) - (C3)$ is called a *carefully synchronizing word* for $\mathcal{A}$. Thus, when a carefully synchronizing word is applied at any state in Q, no undefined transition occurs during the course of application.

If a word w satisfies the condition $(C3)$, it is called an *exactly synchronizing word* for $\mathcal{A}$. Thus, w can be undefined on some states in Q but there must be a state at which w is defined and $q.w = q'.w$ whenever w is defined at any $q, q' \in Q$. Clearly, a carefully synchronizing word is exactly synchronizing but the converse needs not be true. A PFA is said to be *exactly synchronizing* if it possesses an exactly synchronizing word. The class of exactly synchronizing PFAs is much larger than that of carefully synchronizing PFAs: for instance, if one adds to a synchronizing CFA a new state at which no letter is defined, one gets an exactly synchronizing PFA which is not carefully synchronizing since the condition all $(C1)$ fails for each letter.

Both careful synchronization and exact synchronization have interesting connections and numerous applications. In particular, exact synchronization is relevant in biologically inspired computing where exactly synchronizing words appear under the name "constants" in the study of so-called splicing systems, see [8]. Another cause of interest in exact synchronization is provided by so-called ϵ-machines, important models in the theory of stationary information sources, see [9, 10].

In [11], the present author and Volkov used SAT-solvers for finding shortest carefully synchronizing words in carefully synchronizing PFAs. Here we apply the same approach to exact synchronization. In section 2, we overview known results related to the problems of determining whether or not a given PFA is exactly synchronizing and of finding shortest exactly synchronizing words in exactly synchronizing PFAs. In Section 3 we show how instances of the latter problem can be efficiently encoded into instances of the Boolean satisfiability problem (SAT). Section 4 presents some of our experimental results.

2. Testing Exact Synchronization

A PFA $\mathcal{A} = \langle Q, \Sigma \rangle$ is said to be *strongly connected* if for every pair $(q, q') \in Q \times Q$, there exists a word $w \in \Sigma^*$ such that $q' = q.w$. It turns out that for strongly connected PFAs, exact synchronization behaves similarly to synchronization of CFAs. In particular, checking whether a given strongly connected PFA is exactly synchronizing can be done in polynomial time, and for strongly connected exactly synchronizing PFAs with n states, there exits a cubic in n upper bound on the minimum length of exactly synchronizing words. Both these facts are straight forward consequences of the following observation by Travers and Crutchfield [9]:

Proposition 1. A strongly connected PFA $\mathcal{A} = \langle Q, \Sigma, \delta \rangle$ is exactly synchronizing if and only if for every pair $(q, q') \in Q \times Q$ there exists a word w such that either $q.w = q'.w$ or w is defined at one of the states q and q' and undefined at the other.

Unfortunately, sufficiency in Proposition 1 does not extend to general PFAs. In the absence of strong connectivity, properties of exact synchronization became rather similar to those of careful synchronization. The following facts were established by Berlinkov [12, Corollary 1]:

Proposition 2. Testing a given PFA with 2 *input letters for exact synchronization is PSPACE-complete. There is a series of* n*-state PFAs with* 2 *input letters with the minimum length of exactly synchronizing words of magnitude* $2^{\Omega(n)}$.

Due to the complexity of the problems related to exact synchronization, one has to invoke some tools that have proved to be efficient for dealing with computationally hard problems. In this paper we use a satisfiability solver as such a tool.

3. Encoding

For completeness, recall the formulation of the Boolean satisfiability problem (SAT). An instance of SAT is a pair (V, C), where V is a set of Boolean variables and C is a collection of clauses over V . (A *clause* over V is a disjunction of literals and a *literal* is either a variable in V or the negation of a variable in V.) Any *truth assignment* on V, i.e., any map $\varphi: V \to \{0,1\}$, extends to a map $C \to \{0,1\}$ (still denoted by φ) via the usual rules of propositional calculus: $\varphi(\neg x) := 1 - \varphi(x), \varphi(x \vee y) := \max\{\varphi(x), \varphi(y)\}$. A truth assignment φ *satisfies* C if $\varphi(c) = 1$ for all $c \in C$. The answer to an instance (V, C) is YES if (V, C) has a *satisfying assignment* (i.e., a truth assignment on V that satisfies C) and NO otherwise.

Nowadays, many efficient *SAT solvers*, i.e., specialized programs designed to solve instances of SAT, have become available. Modern SAT solvers handle instances of SAT with hundreds of thousands of variables and millions of clauses within a few minutes. This remarkable progress has given rise to the following approach to computationally hard problems: one encodes instances of such problems into instances of SAT that are then fed to a SAT-solver. Here we apply this approach to the following problem:

ESW (the existence of an exactly synchronizing word of a given length):
Input: a PFA $\mathcal{A}$ and a positive integer ℓ (given in unary);
Output: YES if $\mathcal{A}$ has an exactly synchronizing word of length ℓ; NO otherwise.

We have to assume that the integer ℓ is given in unary because with ℓ given in binary, an efficient reduction from ESW to SAT is hardly possible in view of Proposition 2.

Now, given an arbitrary instance $(\mathcal{A}, \ell)$ of ESW, we construct an instance (V, C) of SAT such that the answer to $(\mathcal{A}, \ell)$ is YES if and only if so is the answer to (V, C). Here we use the same set of variables as in [11] but the set of clauses is essentially different. One may think that since the definition of an exactly synchronizing word is obtained from the definition of a carefully synchronizing word by omitting the conditions $(C1)$ and $(C2)$, one can get an encoding for ESW by just omitting groups of clauses that control these conditions in the encoding used in [11]. However, it is easy to exhibit counterexamples to show that this naive approach fails.

So, take a PFA $\mathcal{A} = \langle Q, \Sigma \rangle$ and an integer $\ell > 0$. Let $n := |Q|$ and $m := |\Sigma|$and fix some numbering of Q and Σ so that $Q = \{q_1, \dots, q_n\}$ and $\Sigma = \{a_1, \dots, a_m\}$.

The set V of variables in our encoding of $(\mathcal{A}, \ell)$ consists of $m\ell$ *letter variables* $x_{i,t}$ with $1 \le i \le m$, $1 \le t \le \ell$, and $n(\ell + 1)$ *state variables* $y_{j,t}$ with $1 \le j \le n$, $0 \le t \le \ell$. The letter variables encode the letters of a hypothetical exactly synchronizing word w of length ℓ: namely, we want the value of the variable $x_{i,t}$ to be 1 if and only if the t-th letter of w is a_i. The intended meaning of the state variables is as follows: we want the value of the variable $y_{j,t}$ to be 1 whenever the state q_j belongs to the image of Q under the action of the prefix of w of length t, in which situation we say that q_j *is active after* t *steps*. We see that the total number of variables in V is $m\ell + n(\ell + 1) = (m + n)\ell + n$.

The set C of clauses in the encoding of $(\mathcal{A}, \ell)$ consists of four groups. The group I of *initial clauses* contains n one-literal clauses $y_{j,0}$, $1 \le j \le n$, and expresses the fact that all states are active after 0 steps.

For each $t = 1, \dots, \ell$, the group L of *letter clauses* includes the clauses

$$x_{1,t} \vee \dots \vee x_{m,t}, \quad \neg x_{r,t} \vee \neg x_{s,t}, \text{ where } 1 \le r < s \le m. \tag{1}$$

Clearly, the clauses (1) express the fact that the t-th position of our hypothetical exactly synchronizing word w is occupied by exactly one letter in Σ. Altogether, L contains $\ell\left(\frac{m(m-1)}{2} + 1\right)$ clauses.

The next group of clauses encodes the transitions of $\mathcal{A}$. For a state $q_j \in Q$, let $P_i(q_j)$ stand for the set of all preimages of q_j under the action of the letter a_i, that is,

$$P_i(q_j) \coloneqq \{p \in Q \mid p \,.\, a_i = q_j\}.$$

Consider for every $t = 1, \dots, \ell$ and every $j = 1, \dots, n$, the following formula:

$$y_{j,t} = \bigvee_{i=1}^{m} \left(x_{i,t} \& \bigvee_{q_k \in P_i(q_j)} y_{k,t-1}\right). \tag{2}$$

Observe that the equivalence (2) just expresses, in the language of propositional logic, the fact that the state q_j is active after t steps if and only if some preimage of q_j under the action of the t-th letter of w is active after $t-1$ steps. A direct conversion of (2) into a conjunctive normal form leads to quite a bulky system of clauses. Instead, we use the following lemma.

Lemma 3. Fix numbers $t \in \{1, \dots, \ell\}$ and $j = \{1, \dots, n\}$ and take any truth assignment $\varphi: V \to \{0,1\}$ such that $\varphi(x_{i,t}) = 1$ for exactly one value of $i \in \{1, \dots, m\}$.. Then φ satisfies the equivalence (2) *if and only if φ satisfies the following system of clauses:*

$$\neg y_{j,t} \vee \neg x_{i,t} \vee \bigvee_{q_k \in P_i(q_j)} y_{k,t-1} \quad \text{for each } i \in \{1, \dots, m\}, \tag{3}$$

$$y_{j,t} \vee \neg x_{i,t} \vee \neg y_{k,t-1}, \quad \text{for each } i \in \{1, \dots, m\} \text{ and each } q_k \in P_i(q_j). \tag{4}$$

Proof. Let i_0 be such that $\varphi(x_{i_0}, t) = 1$. Then $\varphi(x_i, t) = 0$ for all $i \neq i_0$ whence the right-hand side of the equivalence (2) gets the same value under φ as the expression $\bigvee_{q_k \in P_{i_0}(q_j)} y_{k,t-1}$. Thus, if (2) is satisfied by φ, so are the implications

$$y_{j,t} \to \bigvee_{q_k \in P_{i_0}(q_j)} y_{k,t-1}, \tag{5}$$

$$\bigvee_{q_k \in P_{i_0}(q_j)} y_{k,t-1} \to y_{j,t}. \tag{6}$$

The implication (5) is equivalent to the clause $\neg y_{j,t} \vee \bigvee_{q_k \in P_{i_0}(q_j)} y_{k,t-1}$, and we see that φ satisfies the clause (3) with $i = i_0$. The implication (6) is equivalent to the system of clauses $y_{j,t} \vee \neg y_{k,t-1}$ where k runs over the indices of states in $P_{i_0}(q_j)$. Hence, if φ satisfies (6), we see that φ satisfies all clauses (4) with $i = i_0$. Besides that, φ satisfies all clauses (3) and (4) with $i \neq i_0$ because each of these clauses includes $\neg x_{i,t}$ as a literal and $\varphi(x_{i,t}) = 0$ for all $i \neq i_0$.

Thus, we have shown that if φ satisfies the equivalence (2), then φ also satisfies all clauses (3) and (4). All our arguments are reversible, and therefore, the converse claim holds as well.

We collect clauses of the form (3) and (4) for all $t = 1, \dots, \ell$ and $j = 1, \dots, n$ in the group T of *transition clauses*. There are ℓmn clauses of the form (3); as for clauses of the form (4), its number for each triple (i, j, t) depends on the cardinality of the set $P_i(q_j)$, which clearly does not exceed n. Hence the number of clauses of the form (4) does not exceed ℓmn^2 whence T contains at most $\ell mn(n+1)$ clauses in total.

It may be worth explaining how the clauses of the form (3) and (4) are understood in the case when one of the sets $P_i(q_j)$ happens to be empty. In (3) the disjunction over the empty set is omitted so that if $P_i(q_j) =$

ϕ, then the clause (3) reduces to $\neg y_{j,t} \vee \neg x_{i,t}$. The latter clause simply means that if the t-th letter of w is a_i and the state q_j has no preimage under a_i, then q_j cannot be active after t steps. As for (4), the clauses of this sort disappear for all i such that $P_i(q_j)$ is empty.

The final group S of *synchronization clauses* describes the situation at the end of the synchronization process when the action of the word w is completed. It consists of the following clauses:

$$y_{1,\ell} \vee \ldots \vee y_{n,\ell}, \qquad \neg y_{r,\ell} \vee \neg y_{s,\ell} \quad \text{where } 1 \le r < s \le n. \tag{7}$$

Clearly, the clauses in (7) express the fact that that exactly one state remains active after ℓ steps, which corresponds to the requirement $|Q.w| = 1$. The group S contains $(\frac{n(n-1)}{2} + 1)$ clauses.

We let $C \coloneqq \mathrm{I} \cup L \cup T \cup S$. The number of clauses in the set C is no greater than $\ell m \left(\frac{m+2n(n+1)-1}{2}\right) + (\ell + 1) + \frac{n(n+1)}{2}$. Now we can state and prove the main theorem of this section.

Theorem 4. A PFA $\mathcal{A}$ has an exactly synchronizing word of length ℓ if and only if the instance (V, C) of SAT constructed above is satisfiable, and the construction of this instance can be done in polynomial time. Moreover, the exactly synchronizing words of length ℓ for $\mathcal{A}$ are in a one-to-one correspondence with the satisfying assignments of (V, C).

Proof. We keep the notation and terminology introduced in the course of the construction of our encoding $(\mathcal{A}, \ell) \to (V, C)$. In particular, $\mathcal{A} = \langle Q, \Sigma \rangle$ with $Q = \{q_1, \ldots, q_n\}$ and $\Sigma = \{a_1, \ldots, a_m\}$. The fact that the instance (V, C) can be constructed in polynomial of n, m, and ℓ time follows from the estimates of $|V|$ and $|C|$ established above.

Now suppose that $\mathcal{A}$ has an exactly synchronizing word of length ℓ and fix such a word w. We define a truth assignment $\varphi: V \to \{0,1\}$ as follows: for the letter variables $x_{i,t}$ with $1 \le i \le m$, $1 \le t \le \ell$,

$$\varphi(x_{i,t}) = \begin{cases} 1 & \text{if the } t^{\text{th}} \text{ letter of w is } \mathrm{a_i}, \\ 0 & \text{otherwise.} \end{cases}$$

for the state variables $y_{j,t}$ with $1 \le j \le n$, $0 \le t \le \ell$,

$$\varphi(y_{j,t}) = \begin{cases} 1 & \text{if the state } q_j \text{ is active after } t \text{ steps}, \\ 0 & \text{otherwise.} \end{cases}$$

Clearly, φ satisfies all clauses in L and I. Since w is an exactly synchronizing word, exactly one state remains active after ℓ steps, whence φ satisfies all clauses in S. By the construction, φ satisfies the formulas (2) for $t = 1, \ldots, \ell$ and $j = 1, \ldots, n$ as these formulas describe the propagation of active states. Since φ is such that $\varphi(x_{i,t}) = 1$ for exactly one value of $i \in \{1, \ldots, m\}$, Lemma 3 ensures that φ also satisfies all clauses in T. We see that the SAT-instance (V, C) is satisfied by φ.

Conversely, suppose that (V, C) is satisfiable and fix a satisfying assignment φ for (V, C). Since φ satisfies all clauses in L, for each $t = 1, \ldots, \ell$, there exists exactly one index $i(t) \in \{1, \ldots, m\}$ such that $\varphi(x_{i(t),t}) = 1$. Define a word $w \coloneqq a_{i(1)} \ldots a_{i(\ell)}$ and let w_t be the prefix of w of length t for $t = 1, \ldots, \ell$. (Here w_0 is the empty word ε.) We aim to prove that for each $t = 0, 1, \ldots, \ell$,

$$Q.w_t = \{q_j \mid \varphi(y_{j,t}) = 1\}. \tag{8}$$

We induct on t. The claim (8) holds for $t = 0$ since φ satisfies all clauses in I. Now suppose that $t > 0$. Lemma 3 applies to φ whence the condition that φ satisfies all clauses in I implies that φ also satisfies the equivalencies (2) for $t = 1, \dots, \ell$ and j= $1, \dots, n$. By the induction assumption, we have that a state q_k is active after $t-1$ steps if and only if $\varphi(y_{k,t-1}) = 1$. As mentioned, the equivalence (2) translates into the language of propositional logic that q_j is active after t steps if and only if some preimage of q_j under the action of the letter $a_{i(t)}$ is active after $t-1$ steps; on the other hand, since φ satisfies (2), we have $\varphi(y_{j,t}) = 1$ if and only if $\varphi(y_{k,t-1}) = 1$ for some $k \in P_{i(t)}(q_j)$. Combining these two facts, we get (8).

Since φ satisfies all clauses in S, the equality $\varphi(y_{j,\ell}) = 1$ holds for exactly one value of $j \in \{1, \dots, n\}$. By (8), this means that $|Q.w_\ell| = 1$, that is, $w = w_\ell$ is an exactly synchronizing word for $\mathcal{A}$.

We see that any exactly synchronizing word of length ℓ for $\mathcal{A}$ determines a satisfying assignment for (V, C) and vice verse. Moreover, from the above proof it is clear that if we start with an exactly synchronizing word w of length ℓ, construct from w a satisfying assignment φ for (V, C), and then build an exactly synchronizing word from φ, we get back the word w. Thus, the correspondence between the exactly synchronizing words of length ℓ for $\mathcal{A}$ and the satisfying assignments of (V, C) is indeed one-to-one.

4. Experimental results

In this section we present some of our experimental results. The main aim of these experiments is the estimation of the expected length of the shortest exactly synchronizing word for PFAs with two input letters, n states, and precisely one undefined transition. In [11] a similar series of experiments have been performed for careful synchronization, and it appears to be natural to look at similarities and differences between the two versions of synchronization.

As in [11], the number n in our series of experiments did not exceed 100. We have generated uniformly at random a sample of 5000 automata for each $n \leq 20$, 2000 automata for $25 \leq n \leq 60$, and 700 automata for $65 \leq n \leq 100$. In order to find an exactly synchronizing word of minimum length for generated PFAs, we have considered ESW instances $(\mathcal{A}, \ell)$ with each $\mathcal{A}$ in the sample and applied the encoding constructed in Section 3 to solve ESW instances with the help of a SAT solver. Then the minimum length been calculated via binary search on ℓ; we refer the reader to [13, 14] where the procedure of binary search on ℓ is presented in detail. We have used MiniSat 2.2.0 [15, 16]; the algorithm outlined above has been implemented in C++ and the code has been compiled with GCC 4.9.2. In our experiments we have used a personal computer with an Intel(R) Core(TM) i5-2520M processor with 2.5 GHz CPU and 4GB of RAM.

For each n, we have calculated the average length $\ell(n)$ of shortest exactly synchronizing words for generated PFAs. Finally, the least squares method has been used to find a function that best reflects how $\ell(n)$ depends on n, and it turned out that our results are reasonably well approximated by the following expression:

$$\ell(n) \approx 0.12 + 0.44n - 0.004n^2 + 0.000018n^3 \quad (9)$$

The relation between this approximation and our experimental data is shown in figure 1.

In figure 2, we see that the relative standard deviation of $\ell(n)$ gradually decreases as the number of states grows.

Finally, we compare our experimental results with those of [11] where the average length of shortest carefully synchronizing words for the same class of PFAs (PFAs with two input letters and precisely one undefined transition) has been evaluated. As expected, the average length of shortest exactly synchronizing words turns out to be smaller than the average length of shortest carefully synchronizing words−recall that every carefully synchronizing word is exactly synchronizing but the converse needs not be true. However, both values seem to follow the same pattern at least for the special class of PFAs used in our experiment. These observations are illustrated in figure 3.

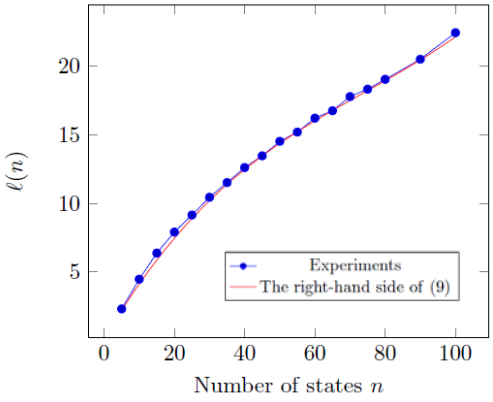

Figure 1. Experimental data vs. approximation in (9).

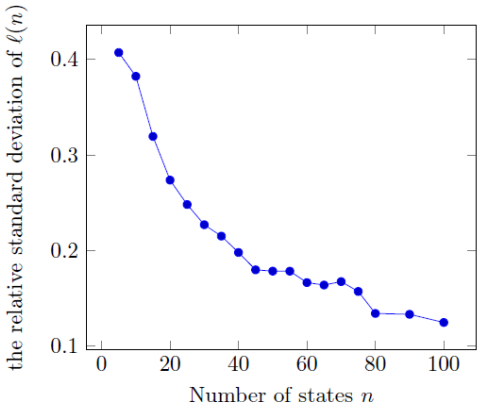

Figure 2. The relative standard deviation of $\ell(n)$.

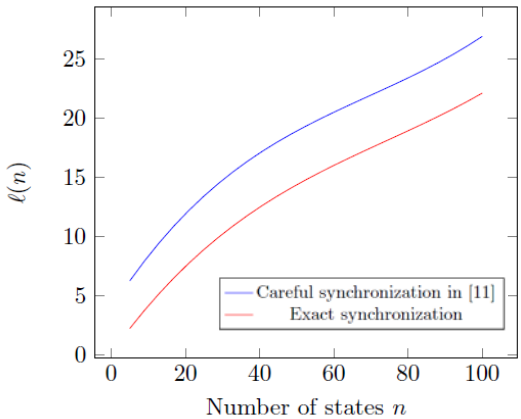

Figure 3. Comparison between exact and careful synchronization.

5. Conclusion

We have approached the problem of computing an exactly synchronizing word of minimum length for a given PFA via the SAT solver method. For this, we have developed a new encoding that essentially differs from ones used in the earlier papers by the author [13] and by the author and Volkov [11, 14]. We have implemented our algorithms and performed a number of experiments that confirm that our approach works sufficiently well even with very basic SAT solver and very modest computational resources.

Now we are designing new experiments. For instance, it appears to be interesting to compare the minimum lengths of exactly synchronizing words for exactly synchronizing PFAs with a fixed number of states and letters but with different number of transitions.

References

[1] Volkov M V 2008 Synchronizing automata and the Cerny conjecture *Languages and Automata Theory and Applications, 2nd Int. Conf., LATA* (Lect. Notes Comput. Sci. Vol **5196**) ed C Martın-Vide, F Otto et al (Berlin: Springer) 11–27

[2] Kari J and Volkov M V 2019 Cerny's conjecture and the Road Coloring Problem *Handbook of Automata Theory* Vol **1** ed J-E Pin (Zürich: EMS Publishing House) 15

[3] Martyugin P V 2008 Lower bounds for the length of the shortest carefully synchronizing words for two- and three-letter partial automata *Diskretn. Anal. Issled. Oper*. **15 (4)** 44–56

[4] Martyugin P V 2010 A lower bound for the length of the shortest carefully synchronizing words *Russian Math.(Iz. VUZ)* **54 (1)** 46–54

[5] Martyugin P V 2012 Synchronization of automata with one undefined or ambiguous transition *Implementation and Application of Automata, 17th Int. Conf., CIAA* (Lect. Notes Comput. Sci. Vol **7381**) ed N Moreira and R Reis (Berlin: Springer) 278–288

[6] Martyugin P V 2013 Careful synchronization of partial automata with restricted alphabets *Computer Science – Theory and Applications, 8th Int. Comp. Sci. Symp. in Russia, CSR 2013* (Lect. Notes Comput. Sci. Vol **7913**) ed A A Bulatov and A M Shur (Berlin: Springer) 76–87

[7] Martyugin P V 2014 Complexity of problems concerning carefully synchronizing words for PFA and directing words for NFA *Theory Comput. Syst* **54 (2)** 293–304

[8] Bonizzoni P and Jonoska N 2015 Existence of constants in regular splicing languages *Information and Computation* **242** 340–353

[9] Travers N and Crutchfield J 2011 Exact synchronization for finite-state sources *J. Stat. Phys* **145 (5)** 1181–1201

[10] Travers N and Crutchfield J 2011 Asymptotic synchronization for sinite-state sources *J. Stat. Phys*. **145 (5)** 1202–1223

[11] Shabana H and Volkov M V 2019 Using Sat solvers for synchronization issues in partial deterministic automata Math. *Optimization Theory and Operation Research, MOTOR 2019* (Comm. Comput. Information Sci.) ed M Yu Khachay (Berlin: Springer) (*Preprint* arxiv.org/abs/1903.10549)

[12] Berlinkov M V 2014 On two algorithmic problems about synchronizing automata *Developments in Language Theory – 18th Int. Conf., DLT 2014* (Lect. Notes Comput. Sci. Vol **8633**) ed A M Shur and M V Volkov (Berlin: Springer) 61–67

[13] Shabana H 2018 D2-synchronization in nondeterministic automata *Ural Math. J*. **4 (2)** 99–110

[14] Shabana H and Volkov M V 2018 Using Sat solvers for synchronization issues in nondeterministic automata, *Siberian Electronic Math. Reports* **15** 1426–1442

[15] Een N and Sörensson N 2004 An extensible SAT-solver *Theory and Applications of Satisfiability Testing, 6th Int. Conf., SAT* (Lect. Notes Comput. Sci. vol **2919**) ed E Giunchiglia and A Tacchella (Berlin: Springer) 502–518

[16] Een N and Sörensson N The MiniSat Page http://minisat.se (accessed 18.07.2019)